Passerelle - Development #55230

toulouse_smart: endpoint POST Intervention

28 juin 2021 14:56 - Nicolas Roche

Statut:	Fermé	**Début:**	28 juin 2021
Priorité:	Normal	**Echéance:**	
Assigné à:	Nicolas Roche	**% réalisé:**	0%
Catégorie:		**Temps estimé:**	0:00 heure
Version cible:			
Patch proposed:	Oui	**Planning:**	Non

Description

Ajouter le endpoint POST Intervention (à minima) au connecteur smart.

Demandes liées:

Lié à Passerelle - Bug #55840: Pouvoir tenter un envoi synchrone avant de lan...	**Rejeté**	**26 juillet 2021**

Révisions associées

Révision 813bfff7 - 06 août 2021 14:44 - Nicolas Roche

toulouse_smart: manage status code in mock_response (#55230)

Révision 3ae9b864 - 06 août 2021 14:44 - Nicolas Roche

toulouse_smart: add get-intervention endpoint (#55230)

Révision ff14f2c0 - 06 août 2021 14:44 - Nicolas Roche

toulouse_smart: allow several queries in mock_response (#55230)

Révision 5c6a50af - 06 août 2021 14:44 - Nicolas Roche

toulouse_smart: manage payload and exception in mock_response (#55230)

Révision ba6f3523 - 06 août 2021 14:44 - Nicolas Roche

toulouse_smart: add create-intervention endpoint (#55230)

Révision 8e477662 - 06 août 2021 14:44 - Nicolas Roche

wcs: allow post_json to ask for json content (#55230)

Révision 9204ba61 - 06 août 2021 14:44 - Nicolas Roche

toulouse_smart: add update-intervention endpoint (#55230)

Historique

#3 - 01 juillet 2021 15:31 - Benjamin Dauvergne

Concernant le endpoint POST je suis d'avis de prendre le formulaire en entier et de faire le mapping coté passerelle entre les noms de variale w.c.s. et ceux de SMART, mais il faut d'abord récupérer le schéma complet du formulaire et se mettre d'accord sur un schéma stable coté w.c.s. normalement on devrait obtenir le type d'intervention SMART depuis la fiche de nomenclature AT. Je dois mettre à jour passerelle pour avoir la source de donnée mais on pourrait déjà pré-créer le champ correspondant. Une fois qu'on a le type d'intervention, on prend le slug est ça doit correspondable à un champ bloc dans ce qui a été posté, ou bien erreur.

On poste ça, on obtient une uuid et on le garde dans une table en correspondance avec le form_number ainsi on peut identifier les rejeux.

Comme notificationUrl on file une URL sur passerelle contenant l'id de la ligne de la table de correspondance, ainsi on peut notifier w.c.s. sur l'URL de trigger qu'on voudra (on peut bêtement stocker form_url, et ensuite envoyer vers form_url + /trigger/smart).

Il y a aussi le point des fichiers attachés pour lequel je n'ai pas la liste des champs coté formulaire et aussi ce qu'il faut en faire à part les ajouter à la demande d'intervention (est-ce qu'il y a un ordre ou une nomenclature ou est-ce que c'est juste une liste de fichiers attachés sans structure ?).

#4 - 01 juillet 2021 17:43 - Benjamin Dauvergne

Concernant les notifications je suis d'avis de les stocker localement dans une table et d'appeler le trigger de manière asynchrone, jusqu'à qu'il réponde 200, puis de passer la notification dans un statut erreur et de logger cette erreur au bout d'un certain temps.

#5 - 09 juillet 2021 17:39 - Nicolas Roche

- Statut changé de Nouveau à En cours

> Concernant le endpoint POST je suis d'avis de prendre le formulaire en entier et de faire le mapping coté passerelle entre les noms de variale w.c.s. et ceux de SMART, mais il faut d'abord récupérer le schéma complet du formulaire et se mettre d'accord sur un schéma stable coté w.c.s.

Via l'option "Envoyer les données du formulaire" dans l'action du WS ?
C'est pas un peu violent ?

> normalement on devrait obtenir le type d'intervention SMART depuis la fiche de nomenclature AT. Je dois mettre à jour passerelle pour avoir la source de donnée mais on pourrait déjà pré-créer le champ correspondant.

Tu parles d'ajouter aux blocs de champs un champ calculé contenant (le type d'intervention SMART) ?

> Il y a aussi le point des fichiers attachés pour lequel je n'ai pas la liste des champs coté formulaire et aussi ce qu'il faut en faire à part les ajouter à la demande d'intervention (est-ce qu'il y a un ordre ou une nomenclature ou est-ce que c'est juste une liste de fichiers attachés sans structure ?).

J'ai l'impression que ce sera géré indépendamment, dans un second temps, via /v1/intervention/{interventionId}/media (Ajout d'un média)
genre (mais qui ne marche pas encore):

```
$ curl -H 'Content-Type: image/jpeg' https://admin:password@passerelle.cutm-publik-preprod.nfrance.com/smart/v1/intervention/1c7cbbd0-cc4c-46a7-9ec0-d8b2c5a38037/media -T ~/Images/Kamoulox.jpg
```

> Concernant les notifications je suis d'avis de les stocker localement dans une table et d'appeler le trigger de manière asynchrone, jusqu'à qu'il réponde 200, puis de passer la notification dans un statut erreur et de logger cette erreur au bout d'un certain temps.

Zut, je suis parti sur un endpoint update-intervention côté passerelle.
C'est pas plus simple de laisser la gestion des erreurs à Smart ?
Il avait l'air volontaire.

#6 - 09 juillet 2021 21:29 - Nicolas Roche

- Fichier form-smart.wcs ajouté

- Fichier workflow-smart.wcs ajouté

- Tracker changé de Support à Bug

- Statut changé de En cours à Solution proposée

- Patch proposed changé de Non à Oui

Pour tester :

- get-intervention

```
$ curl 'https://passerelle.dev.publik.love/toulouse-smart/test/get-intervention?id=1c7cbbd0-cc4c-46a7-9ec0-d8b2c5a38037'
```

- post-intervention (via le formulaire/workflow)
- update-intervention

```
$ curl -X POST 'https://passerelle.dev.publik.love/toulouse-smart/test/update-intervention?id=1c7cbbd0-cc4c-46a7-9ec0-d8b2c5a38037&trigger=goto4'
```

#7 - 09 juillet 2021 22:00 - Nicolas Roche

- Fichier 0007-toulouse_smart-add-update-intervention-endpoint-5523.patch ajouté

- Fichier 0006-toulouse_smart-add-create-intervention-endpoint-5523.patch ajouté

- Fichier 0005-toulouse_smart-check-payload-in-mock_response-55230.patch ajouté

- Fichier 0004-toulouse_smart-allow-several-queries-in-mock_respons.patch ajouté

- Fichier 0003-toulouse_smart-move-make_id-into-utils-55230.patch ajouté

- Fichier 0002-toulouse_smart-add-get-intervention-endpoint-55230.patch ajouté

- Fichier 0001-toulouse_smart-manage-status-code-in-mock_response-5.patch ajouté

#8 - 10 juillet 2021 18:06 - Benjamin Dauvergne

Nicolas Roche a écrit :

> Concernant le endpoint POST je suis d'avis de prendre le formulaire en entier et de faire le mapping coté passerelle entre les noms de variale w.c.s. et ceux de SMART, mais il faut d'abord récupérer le schéma complet du formulaire et se mettre d'accord sur un schéma stable coté w.c.s.

Via l'option "Envoyer les données du formulaire" dans l'action du WS ?
C'est pas un peu violent ?

Ça me semble le plus simple au contraire, sinon l'appel de web-service va être diablement compliqué avec des conditions partout. C'est plus simple de respecter une politique de nommage coté w.c.s. et d'envoyer tout. On pourrait poser que pour le type d'intervention "xxx" qui a un champ spécifique "yyy" il faut un champ bloc de champ nommé "intervention_xxx" qui contient un sous-champ avec pour nom de variable "yyy". Coté passerelle on cherchera donc dans le payload une variable "intervention_xxx_yyy" (si je ne me trompe pas).

> normalement on devrait obtenir le type d'intervention SMART depuis la fiche de nomenclature AT. Je dois mettre à jour passerelle pour avoir la source de donnée mais on pourrait déjà pré-créer le champ correspondant.

Tu parles d'ajouter aux blocs de champs un champ calculé contenant ?

Non, on va avoir un champ liste dans le formulaire qui pointe sur la nomenclature AT (des fiches), disons form_var_nomenclature_at. Ces fiches auront un champ "type d'intervention SMART" contenant le slug du type d'intervention (slugify du nom en attendant que SMART ait un identifiant stable à nous proposer), ce sera form_var_nomenclature_at_type_intervention_smart, il faudra envoyer ce champ en champ supplémentaire au connecteur (il se retrouvera dans {"extra": {"type_intervention": ...}} vu qu'on envoie la totalité du formulaire, il faudrait prendre ne compte ce champ "extra", simplement en faisant payload.update(payload.pop("extra", {})) si passerelle ne le fait pas déjà tout seul).

> Il y a aussi le point des fichiers attachés pour lequel je n'ai pas la liste des champs coté formulaire et aussi ce qu'il faut en faire à part les ajouter à la demande d'intervention (est-ce qu'il y a un ordre ou une nomenclature ou est-ce que c'est juste une liste de fichiers attachés sans structure ?).

J'ai l'impression que ce sera géré indépendamment, dans un second temps, via /v1/intervention/{interventionId}/media (Ajout d'un média) genre (mais qui ne marche pas encore):
[...]

Justement j'avais envie de ne pas reproduire l'API de SMART coté connecteur, d'avoir un seul appel demande+fichiers pour simplifier l'usage depuis w.c.s., à charge pour le job de gérer l'état (création intervention, envoie pièce jointe 1, 2, 3, etc..).

Il faut une table demande d'intervention sur le connecteur + une table fille pour les pièce jointes; une fois la demande créée on pose un job pour créer la demande dans SMART.

Dans le job une fois que la demande est créé (sinon rejeu) on stocke l'UUID dans l'instance demande d'intervention puis on lance n jobs pour poser les n pièces. jointes. Idem dans les jobs pour les pièces jointes, on rejoue tant qu'on a pas réussi à pousser la pièce jointe.

> Concernant les notifications je suis d'avis de les stocker localement dans une table et d'appeler le trigger de manière asynchrone, jusqu'à qu'il réponde 200, puis de passer la notification dans un statut erreur et de logger cette erreur au bout d'un certain temps.

Zut, je suis parti sur un endpoint update-intervention côté passerelle.
C'est pas plus simple de laisser la gestion des erreurs à Smart ?
Il avait l'air volontaire.

Quelle gestion d'erreur ? Ils ont dit qu'il feraient une notification, on ne sait pas s'ils vont gérer le rejeu, on a déjà du mal à avoir une API correcte. Perso je vois le schéma suivant :

```
SMART --- notification --> {{passerelle_url}}/toulouse_smart/notification/<id-demande-intervention-dans-passer
elle>/ --> pose un job pour notifier w.c.s.
-- plus tard --
CRON --- job notification w.c.s. --> {{wcs_url}}/backoffice/.../allo/trigger/notify
```

Où {{passerelle_url}}/toulouse_smart/notification/<id-demande-intervention-dans-passerelle>/ est l'URL envoyée à SMART lors de la création de la demande dans SMART.

#9 - 12 juillet 2021 09:12 - Nicolas Roche

- Statut changé de Solution proposée à En cours

Ok, je pensais m'en sortir en ne faisant que le minimum syndical côté passerelle, mais là je comprend que c'est tout le couplage wcs+passerelle qu'il me faut considérer.

> mais il faut d'abord récupérer le schéma complet du formulaire et se mettre d'accord sur un schéma stable coté w.c.s.

Comment je peux faire pour monter un formulaire de test ?
Plutôt que tu m'expliques, est-ce que tu pourrais m'en fournir un stp ?

> Justement j'avais envie de ne pas reproduire l'API de SMART coté connecteur, d'avoir un seul appel demande+fichiers pour simplifier l'usage depuis w.c.s., à charge pour le job de gérer l'état (création intervention, envoie pièce jointe 1, 2, 3, etc..).

C'est dommage, on se retrouve à devoir attendre pour développer le connecteur alors que je me suis déjà engagé à le livrer.

> Quelle gestion d'erreur ?

D'habitude on fait du push-pull parce qu'on veut gérer totalement de notre côté,
bien que dans l'idéal on préférerais faire du push-push, parce que c'est plus logique.
Ici on part sur du push-push, mais avec toute l'intelligence de notre côté.
Ça me parait être 2 fois plus compliqué pour le même résultat.

#10 - 12 juillet 2021 10:43 - Benjamin Dauvergne

Nicolas Roche a écrit :

> Ok, je pensais m'en sortir en ne faisant que le minimum syndical côté passerelle, mais là je comprend que c'est tout le couplage wcs+passerelle qu'il me faut considérer.

À part l'ajout d'un modèle et la gestion d'un job ton code est déjà bon je pense (il faudra peut-être changer 2/3 noms de champs).

> > mais il faut d'abord récupérer le schéma complet du formulaire et se mettre d'accord sur un schéma stable coté w.c.s.
>
> Comment je peux faire pour monter un formulaire de test ?
> Plutôt que tu m'expliques, est-ce que tu pourrais m'en fournir un stp ?

https://demarches-montoulouse.cutm-publik-preprod.nfrance.com/backoffice/forms/117/fields/

> > Justement j'avais envie de ne pas reproduire l'API de SMART coté connecteur, d'avoir un seul appel demande+fichiers pour simplifier l'usage depuis w.c.s., à charge pour le job de gérer l'état (création intervention, envoie pièce jointe 1, 2, 3, etc..).
>
> C'est dommage, on se retrouve à devoir attendre pour développer le connecteur alors que je me suis déjà engagé à le livrer.

Je préfère perdre du temps ici que sur l'écriture du workflow et à l'usage quand ça foirera à moitié sur l'envoie des fichiers. Maintenant je nous vois très bien livrer une v1 sans les pièces jointes; il faut juste faire le modèle pour stocker le contenu de la demande et la pousser dans un job. Dans un deuxième temps on pourra implémenter un deuxième modèle pour stocker les pièces jointes, avec une FK vers le modèle des demandes et gérer leur envoie vers SMART.

> > Quelle gestion d'erreur ?
>
> D'habitude on fait du push-pull parce qu'on veut gérer totalement de notre côté,
> bien que dans l'idéal on préférerais faire du push-push, parce que c'est plus logique.
> Ici on part sur du push-push, mais avec toute l'intelligence de notre côté.
> Ça me parait être 2 fois plus compliqué pour le même résultat.

Je ne comprends pas le lien avec la gestion d'erreur, coté w.c.s. je trouve toujours complexe de gérer le rejeu à la main.

#11 - 12 juillet 2021 11:04 - Benjamin Dauvergne

Benjamin Dauvergne a écrit :

> > Justement j'avais envie de ne pas reproduire l'API de SMART coté connecteur, d'avoir un seul appel demande+fichiers pour simplifier l'usage depuis w.c.s., à charge pour le job de gérer l'état (création intervention, envoie pièce jointe 1, 2, 3, etc..).

> C'est dommage, on se retrouve à devoir attendre pour développer le connecteur alors que je me suis déjà engagé à le livrer.

> Je préfère perdre du temps ici que sur l'écriture du workflow et à l'usage quand ça foirera à moitié sur l'envoie des fichiers. Maintenant je nous vois très bien livrer une v1 sans les pièces jointes; il faut juste faire le modèle pour stocker le contenu de la demande et la pousser dans un job. Dans un deuxième temps on pourra implémenter un deuxième modèle pour stocker les pièces jointes, avec une FK vers le modèle des demandes et gérer leur envoie vers SMART.

Pour le modèle des demandes je vois un truc comme ça :

```
class WcsRequest(models.Model):
    ressource = FK(Ressource)
    uuid = UUIDField(default=uuid.uuid4) <-- pour avoir un nombre aléatoire à utiliser dans l'URL de notificat
ion qui ne devra pas utiliser d'authentification, il faut que ce soit difficilement devinable
    wcs_form_number = CharField(...., max_length=16) # <--- l'id de la demande dans w.c.s. ça permet de retrou
ver le job facilement sur un replay de w.c.s.
    payload = JSONField(...) # <-- ce que w.c.s. a envoyé
    result = JSONField(...., null=True) # <-- ce que SMART a retourné
    status = CharField(..., default='received', max_length=16) # received, sent, etc..
```

#12 - 16 juillet 2021 15:23 - Nicolas Roche

Juste pour info,
lorsque j'utilise l'option 'Envoyer les données du formulaire', le bloc de champ n'est pas encodé en utilisant les identifiants des champs tels qu'ils sont exportés dans le zip.
(du coup je fait le rapprochement sur le nom des champs).

```
(Pdb) json.loads(request.body)['fields']['barrieres_de_police_genantesoubliees_raw']
[
  {
    "prec_localisation_raw": "Ne sait pas",
    "prec_localisation": "Ne sait pas",
    "couleur": "vert canard",
    "nb_estime": "2"
  }
]
```

#13 - 16 juillet 2021 15:33 - Frédéric Péters

> le bloc de champ n'est pas encodé en utilisant les identifiants des champs tels qu'ils sont exportés dans le zip.

Tu peux donner un peu de contexte et dire ce que tu attendais ?

#14 - 16 juillet 2021 16:12 - Nicolas Roche

> et dire ce que tu attendais ?

Je m'attendais à avoir la même structure que lorsque je passais {{ form_var_<slug du bloc> }} en donnée du service web.
J'avais alors :

```
{
  "data": [
    {
      "0b0e0c09-5ae5-0785-fe7a-7b38f882a67e": "2",
      "55e2f822-6831-4dd9-1e7f-0bd71301fd55": "vert cannard",
      "6318a04b-eba8-ea8b-ec72-178ea1aec2df": "Ne sait pas",
      "6318a04b-eba8-ea8b-ec72-178ea1aec2df_display": "Ne sait pas"
    }
  ],
  "schema": {
    "0b0e0c09-5ae5-0785-fe7a-7b38f882a67e": "string",
    "55e2f822-6831-4dd9-1e7f-0bd71301fd55": "string",
    "6318a04b-eba8-ea8b-ec72-178ea1aec2df": "item"
  }
}
```

qui reprennait les identifiants des blocs de l'export du bloc de champ :

```
<?xml version="1.0"?>
<block id="407ef8d6-9b8c-4187-be51-d89f24424768">
  <name>Barrières de police gênantes/oubliées</name>
  <slug>barrieres-de-police-genantesoubliees</slug>
  <fields>
    <field>
      <id>6318a04b-eba8-ea8b-ec72-178ea1aec2df</id>
      <label>PREC_LOCALISATION</label>
      <type>item</type>
      <required>True</required>
      <varname>prec_localisation</varname>
      <display_locations>
        <display_location>validation</display_location>
        <display_location>summary</display_location>
      </display_locations>
      <items>
        <item>Dans rue piétonne</item>
        <item>Ne sait pas</item>
        <item>Sur chaussée</item>
        <item>Sur place</item>
        <item>Sur trottoir</item>
      </items>
    </field>
    <field>
      <id>55e2f822-6831-4dd9-1e7f-0bd71301fd55</id>
      <label>COULEUR</label>
      <type>string</type>
      <required>False</required>
      <varname>couleur</varname>
      <display_locations>
        <display_location>validation</display_location>
        <display_location>summary</display_location>
      </display_locations>
    </field>
    <field>
      <id>0b0e0c09-5ae5-0785-fe7a-7b38f882a67e</id>
      <label>NB_ESTIME</label>
      <type>string</type>
      <required>False</required>
      <varname>nb_estime</varname>
      <display_locations>
        <display_location>validation</display_location>
        <display_location>summary</display_location>
      </display_locations>
      <validation>
        <type>digits</type>
      </validation>
    </field>
  </fields>
</block>
```

À présent, j'ai l'impression que ces identifiants ne serviront plus à rien.
(mais peu m'importe je signal juste cela parce que ça m'a surpris).

#15 - 16 juillet 2021 16:18 - Frédéric Péters

Il ne faut pas, jamais, utiliser ces identifiants internes.

#16 - 16 juillet 2021 18:11 - Benjamin Dauvergne

Frédéric Péters a écrit :

> Il ne faut pas, jamais, utiliser ces identifiants internes.

Oui je n'avais pas dans l'idée qu'on les utiliserait, c'était simplement pour me conformer à la forme générale des champs; je ne savais même pas qu'il y avait moyen de les exporter en JSON de cette façon, c'est bien le varname généré à partir du label qu'il faut utiliser.

#17 - 19 juillet 2021 10:30 - Nicolas Roche

- Fichier 0007-toulouse_smart-add-update-intervention-endpoint-5523.patch ajouté

- Fichier 0006-wcs-allow-post_json-to-ask-for-json-content-55230.patch ajouté

- Fichier 0005-toulouse_smart-add-create-intervention-endpoint-5523.patch ajouté

- Fichier 0004-toulouse_smart-check-payload-in-mock_response-55230.patch ajouté

- Fichier 0003-toulouse_smart-allow-several-queries-in-mock_respons.patch ajouté

- Fichier 0002-toulouse_smart-add-get-intervention-endpoint-55230.patch ajouté

- Fichier 0001-toulouse_smart-manage-status-code-in-mock_response-5.patch ajouté

- Statut changé de En cours à Solution proposée

> Pour le modèle des demandes je vois un truc comme ça : ...

ok (moi j'étais parti pour ré-utiliser l'objet Cache).

> URL de notification qui ne devra pas utiliser d'authentification

ok (moi j'aurais juste utilisé le système d'apikey de passerelle).

0006: j'ai réutilisé le code déjà présent pour appeler les API de w.c.s., mais ce n'est pas tellement plus lisible, surtout lorsqu'il s'agit de choisir entre une réponse négative de w.c.s ou une erreur réseau, où il faut retenter l'appel.

#18 - 22 juillet 2021 10:49 - Benjamin Dauvergne

Dans create-intervention, tu n'a pas séparé l'envoi de la création. Il faudrait retourner immédiatement une réponse à w.c.s. sans le résultat, juste avec l'uuid, envoyer dans un job et dépendre d'une vue get pour le résultat. Comme ça jamais d'erreur à traiter (mais tu peux tenter un envoi synchrone avant de lancer le job par simplicité, comme ça au prochain appel de w.c.s. la réponse est déjà là).

Nicolas Roche a écrit :

> > Pour le modèle des demandes je vois un truc comme ça : ...
>
> ok (moi j'étais parti pour ré-utiliser l'objet Cache).

Tu peux mais ça va te faire gagner 10/20 lignes de code pour un code moins clair à la fin; ça me semble plus simple d'avoir un modèle spécifique surtout quand il va falloir traiter les pièces jointes.

> > URL de notification qui ne devra pas utiliser d'authentification
>
> ok (moi j'aurais juste utilisé le système d'apikey de passerelle).

Je ne sais pas si tu dis ça :

- par souci de simplicité, et dans ce cas non, il suffit de ne pas mettre de "perm" pour avoir une vue publique
- ou de sécurité, et là je ne crois que ça ne rajoute rien au fait d'avoir des URLs unique contenant un uuid que personne ne pourra deviner.

> 0006: j'ai réutilisé le code déjà présent pour appeler les API de w.c.s., mais ce n'est pas tellement plus lisible, surtout lorsqu'il s'agit de choisir entre une réponse négative de w.c.s ou une erreur réseau, où il faut retenter l'appel.

Je ne vois pas le traitement du contenu qu'on va recevoir sur le trigger (en fait je ne sais pas si on reçoit un contenu et lequel ou si on doit faire un get sur la demande, on ne sait rien ici, il faudrait demander).

Par contre le statut updating/updated ne me parait pas signifiant, le modèle demande n'a qu'à garder le statut de remise de la demande initiale; dans le job de mise à jour tu peux juste faire des self.logger.info/warning/error avec l'uuid de la demande.

#19 - 22 juillet 2021 10:49 - Benjamin Dauvergne

- Statut changé de Solution proposée à En cours

#20 - 22 juillet 2021 12:03 - Nicolas Roche

Merci, je revoie ma copie, mais je réagis quand même en espérant gagner un peu de temps.

> Dans create-intervention, tu n'a pas séparé l'envoi de la création.

Oui, je m'y colle (je n'ai pas l'impression que ça ait été dit explicitement).

J'avais un doute sur le fait que des données synchrones auraient du être remontées dans wcs, mais en fait non, ce n'est pas le cas.

> ou de sécurité, et là je ne crois que ça ne rajoute rien au fait d'avoir des URLs unique contenant un uuid que personne ne pourra deviner.

En fait je prend le problème à l'envers :
À considérer que c'est envisageable que smart ajoute apikey=secret dans ses appels au connecteur,
et que ce mécanisme de protection est suffisant,
pourquoi aurait-on besoin d'avoir des URLs uniques contenant un uuid que personne ne pourra deviner ?
(moi j'aurais utilisé directement form_number)

> Je ne vois pas le traitement du contenu qu'on va recevoir sur le trigger

Mince j'ai raté ça.

```
statut smart 'validé
{
  "data": {
     "status":"validé"
  }
}
statut smart 'planifié'
{
  "data": {
     "status":"planifié",
     "date_planification":"2021-12-13"
  }
}
statut smart 'mise en sécurité'
{
  "data": {
     "status":"mise en sécurité",
     "date_mise_en_securite":"2021-12-13"
  }
}
statut smart 'abandonné'
{
  "data": {
     "status":"abandonné",
     "type_retour":"reclassification"
  }
}
statut smart 'close manque info'
{
  "data": {
     "status":"close manque info",
     "type_retour":"reclassification"
  }
}
statut smart 'terminé'
{
  "data": {
     "status":"close manque info",
     "type_retour_cloture":"Smart non Fait",
     "libelle_cloture" : "rien à l'adresse indiquée",
     "commentaire_cloture": "le commentaire"

  }
}
```

> Par contre le statut updating/updated ne me parait pas signifiant,

Je pense que j'en ai besoin pour gérer une potentielle situation d'erreur, ex :

- wcs crée un demande d'intervention dans smart
- smart demande à mettre à jour le statut à statut1
- wcs procrastine mais le connecteur à déjà répondu "oui, chef'
- smart demande à mettre à jour le statut à statut2
- là le connecteur dit :
  'bof, je dois déjà faire passer le connecteur dans le statut1
  et comme je ne gère pas de file des demandes, il va falloir attendre parce que je me suis déjà engagé pour le premier trigger'

C'est exposé plus succinctement par le test test_update_intervention_pending

#21 - 22 juillet 2021 12:21 - Benjamin Dauvergne

Nicolas Roche a écrit :

> Merci, je revoie ma copie, mais je réagis quand même en espérant gagner un peu de temps.
>
> > Dans create-intervention, tu n'a pas séparé l'envoi de la création.
>
> Oui, je m'y colle (je n'ai pas l'impression que ça ait été dit explicitement).

Non mais le principe de base maintenant, je trouve, c'est de toute faire au maximum dans des jobs pour éviter les tickets, "on a reçu une erreur dans w.c.s, que se passe-t-il".

> J'avais un doute sur le fait que des données synchrones auraient du être remontées dans wcs, mais en fait non, ce n'est pas le cas.

On vit dans un monde asynchrone.

> > ou de sécurité, et là je ne crois que ça ne rajoute rien au fait d'avoir des URLs unique contenant un uuid que personne ne pourra deviner.
>
> En fait je prend le problème à l'envers :
> À considérer que c'est envisageable que smart ajoute apikey=secret dans ses appels au connecteur,
> et que ce mécanisme de protection est suffisant,
> pourquoi aurait-on besoin d'avoir des URLs uniques contenant un uuid que personne ne pourra deviner ?
> (moi j'aurais utilisé directement form_number)

Avec des si on mettrait Paris en bouteille, donc là on a une solution dont on est sûr qu'elle va marcher mais tu veux passer par une autre dont tu n'es pas certain que ça va marcher (est-ce que smart peut ajouter ?apikey=moncul, who knows?). Par simplicité pour le debug et les logs si tu veux ajouter form_number=xxxx à l'URL en plus de uuid ça ne me dérange pas (donnée qui sera ignorée par le endpoint mais c'est pas grave, c'est l'uuid qui fait la sécu).

> > Je ne vois pas le traitement du contenu qu'on va recevoir su le trigger
>
> Mince j'ai raté ça.
> [...]

Comme je le dis tu devrais surtout poser la question sur la liste ou dans un ticket pour Cyril.

> > Par contre le statut updating/updated ne me parait pas signifiant,
>
> Je pense que j'en ai besoin pour gérer une potentielle situation d'erreur, ex :
>
> - wcs crée un demande d'intervention dans smart
> - smart demande à mettre à jour le statut à statut1
> - wcs procrastine mais le connecteur à déjà répondu "oui, chef'
> - smart demande à mettre à jour le statut à statut2
> - là le connecteur dit :
>   'bof, je dois déjà faire passer le connecteur dans le statut1
>   et comme je ne gère pas de file des demandes, il va falloir attendre parce que je me suis déjà engagé pour le premier trigger'
>
> C'est exposé plus succinctement par le test test_update_intervention_pending

C'est pas clair mais j'ai l'impression que tu essaies de te protéger de plusieurs notifications venant de SMART qui seraient rejouer dans le désordre par passerelle, si c'est le cas alors il te faut un modèle pour stocker les notifications et les jouer dans l'ordre, je ne vois pas d'autre moyen. Mais on ne peut répondre à cette question que si on sait ce que sont les notifications, si elles ont un contenu, on a effectivement un problème possible, si ça passe par un get-intervention pour obtenir le contenu/statut de la demande dans SMART, ça n'a pas d'importance car on aura de toute façon que le dernier statut de la demande, il faut juste faire en sorte que si on trigger plusieurs fois w.c.s. pour le même statut ça ne fasse rien.

Si jamais la notification contient bien un contenu (statut/message à l'usager) alors tu dois les stocker dans un modèle pour faire une nouvelle queue et les rejouer strictement dans l'ordre dans un job unique par demande, idéalement en lockant la demande pendant le job avec select_for_update() dans une transaction, et une fois la demande lockée tu lis les notifications non envoyées dans l'ordre et tu les pousses à w.c.s.

#23 - 22 juillet 2021 15:15 - Nicolas Roche

Au tout début de https://dev.entrouvert.org/issues/55230#note-18, il reste encore ce point que je ne comprend pas (et qui me fais me dire que je rate

encore quelque-chose) :

> et dépendre d'une vue get pour le résultat.

#24 - 23 juillet 2021 09:46 - Benjamin Dauvergne

Nicolas Roche a écrit :

> Au tout début de https://dev.entrouvert.org/issues/55230#note-18, il reste encore ce point que je ne comprend pas (et qui me fais me dire que je rate encore quelque-chose) :
>
> > et dépendre d'une vue get pour le résultat.

Des bêtises, j'avais oublié les mails un peu obscure de Cyril nous confirmant que la notification contenait bien des données, bien qu'elles soient dans un format bizarre car différent des web-services.

#25 - 26 juillet 2021 12:08 - Benjamin Dauvergne

relecture create_intervention

```
+                data[prop['name']] = cast[prop['type']](block[name])
```

tu fais un cast, faut faire un try/except ValueError et remonter le souci avec le nom du champ et le type attendu.

```
+         self.add_job(
+             'create_intervention_job',
+             wcs_form_number=wcs_request.wcs_form_number,
+         )
```

Ici je passerai directement l'id de l'objet wcs_request, pas wcs_form_number.

En fait je ferai un truc comme ça :

```
try:
    self.create_intervention_job(id=wcs_request.id)
except SkipJob:
        self.add_job(
            'create_intervention_job',
            id=wcs_request.id
        )
```

L'idée étant qu'on fait du synchrone si tout va bien, et sinon on continue en asynchrone (ce serait bien que ce soit une option de add_job

---

```
+         return {'data': {'uuid': wcs_request.uuid}}
```

tu peux retourner tous les champs de wcs_request ici (ainsi que wcs_request.payload) ça mange pas de pain et ça évite d'aller regarder dans passerelle pour débugger.

---

status défini deux fois :

```
+    status = models.CharField(_('Status'), default='received', max_length=64)
+    status = models.CharField(
+        max_length=20,
+        default='received',
+        choices=(
+            ('registered', _('Registered')),
+            ('sent', _('Sent')),
+            ('service-error', _('Service error')),
+            ('wcs-error', _('WCS error')),
+        ),
+    )
```

#26 - 26 juillet 2021 15:23 - Nicolas Roche

> En fait je ferai un truc comme ça :

```
try:
    self.create_intervention_job(id=wcs_request.id)
except SkipJob:
    self.add_job('create_intervention_job', id=wcs_request.id)
```

Bof, en cas d'erreur sur l'appel synchrone, on perd le retour de smart d'enregistré quelque-part pour creuser l'erreur.
En fait j'ai volontairement zappé cette partie en espérant que tu allais me donner la commande homologuée pour lancer les jobs instantanément (afin de ne pas avoir 2 gestions d'erreurs différentes suivant que l'on soit passé en synchrone ou asynchrone.)

> (ce serait bien que ce soit une option de add_job)

ah zut, en fait on essuie les plâtres

edit:
juste appeler jobs() sur le connecteur ça devrait le faire.

#27 - 26 juillet 2021 15:39 - Frédéric Péters

> juste appeler jobs() sur le connecteur ça devrait le faire.

Se prendre le risque d'exécuter n'importe quoi, non.

> > (ce serait bien que ce soit une option de add_job)
>
> ah zut, en fait on essuie les plâtres

Ce qu'il faut ce serait donc un ticket pour avoir ça qui ressemblerait à :

```
@@ -603,7 +603,7 @@ class BaseResource(models.Model):
             if result == 'skipped':
                 skipped_jobs.append(job.id)

-    def add_job(self, method_name, natural_id=None, after_timestamp=None, **kwargs):
+    def add_job(self, method_name, natural_id=None, after_timestamp=None, try_now=False, **kwargs):
         resource_type = ContentType.objects.get_for_model(self)
         job = Job(
             resource_type=resource_type,
@@ -614,6 +614,10 @@ class BaseResource(models.Model):
         )
         job.set_after_timestamp(after_timestamp)
         job.save()
+        if try_now:
+            job.run(spool=False)
+            if job.status == 'completed':
+                return
         transaction.on_commit(lambda: job.run(spool=True))
         return job
```

mais c'est écrit sans tester et il y a peut-être des complexités et très certainement elles apparaitront à la relecture.

#28 - 26 juillet 2021 15:40 - Frédéric Péters

> En fait j'ai volontairement zappé cette partie en espérant que tu allais me donner la commande homologuée pour lancer les jobs instantanément

Et là-dessus, quand de manière consciente tu zappes une demande de la relecture, on gagne du temps si tu l'explicites et en donne la raison quand tu postes le patch qui suit.

#29 - 26 juillet 2021 17:42 - Nicolas Roche

- Lié à Bug #55840: Pouvoir tenter un envoi synchrone avant de lancer le job ajouté

#30 - 26 juillet 2021 18:08 - Nicolas Roche

- Fichier 0007-toulouse_smart-add-update-intervention-endpoint-5523.patch ajouté

- Fichier 0005-toulouse_smart-add-create-intervention-endpoint-5523.patch ajouté

- Statut changé de En cours à Solution proposée

Remarques prises en comptes.
Branche basée sur #55840 (seuls ces 2 patchs ont changé).

#31 - 27 juillet 2021 17:50 - Nicolas Roche

- Fichier 0007-toulouse_smart-add-update-intervention-endpoint-5523.patch ajouté

- Fichier 0005-toulouse_smart-add-create-intervention-endpoint-5523.patch ajouté

Je réalise que ce n'est pas la bonne façon de faire à cause de la transaction sur les jobs qui risque de différer la réponse.
(https://dev.entrouvert.org/issues/55840#note-7)
Je pars donc sur ta proposition (désolé pour le temps perdu) et je recopie les erreurs dans le champ result des deux tables.

#32 - 27 juillet 2021 18:01 - Frédéric Péters

```
data = serializers.serialize('python', [wcs_request])[0]
```

Je n'ai pas tout lu mais vraiment je ne mettrais pas des trucs comme ça; l'appelant il est au courant des paramètres qu'il envoie, pas besoin de les lui répondre. (ou je ne pige pas le sens)

#33 - 27 juillet 2021 19:44 - Nicolas Roche

- Fichier 0007-toulouse_smart-add-update-intervention-endpoint-5523.patch ajouté

- Fichier 0005-toulouse_smart-add-create-intervention-endpoint-5523.patch ajouté

Remarque prise en compte.
J'ai retiré le serializer et laissé seulement les payloads envoyés par le connecteur.

~~Sinon rien à voir mais je me demande :~~
~~> avoir un nombre aléatoire à utiliser dans l'URL de notification qui ne devra pas utiliser d'authentification, il faut que ce soit difficilement devinable~~
~~pour simplifier je devrais peut-être utiliser l'uuid retourné par smart à la création de l'intervention ?~~

edit:
non, c'est plus sûr que ce soit le requêteur qui génère l'id.

#34 - 29 juillet 2021 17:59 - Benjamin Dauvergne

Je vois un paramètre trigger mais je ne sais pas d'où il sort, coté create_intervention il y a juste uuid.

Tu ne peux pas faire .select_for_update(...) en dehors d'une transaction, tu es ici trompé parce que tous les tests s'exécutent dans une transaction, avec la fixture transactional_db tu aurais une erreur (et à l'utilisation ça va planter tout de suite) :

```
+        smarts_requests = SmartRequest.objects.select_for_update().filter(
```

```
+        perm='can_access',
```

on a dit pas d'authent sur le endpoint de notification

Pour l'instant ne te préoccupe pas de l'ordre des rejeux des notifications, c'est un peu trop complexe comment tu fais ça, fais simple on verra plus tard si ça pose un souci (j'en doute) :

```
    @atomic
    def update_intervention_job(self, smart_request_pk):
        smarts_request = SmartRequest.objects.select_for_update().filter(pk=smart_request_pk).first()
        if not smart_request:
            self.logger.error('smart request %s has been lost', smart_request_pk)
            return

        # ok pour le reste
```

--

vire service-error/wcs-errror, tu peux juste laisser le système de job gérer les erreurs, t'as un attribut natural_id sur Job/add_job qui permet de faire le lien éventuellement avec un modèle source (genre natural_id=f'notification-{smart_request.pk}'), je ne suis pas certain que le statut sur le modèle SmartRequest ait un intérêt, c'est le job qui gère ça. En fait tout le bouzin de push/mise à jour de smart_request j'en ferai une méthode de smart_request, dans le job on récupère l'objet on appel smart_request.push() et basta; ça permet de jouer en console ensuite si on veut (SmartRequest.object.get(id=pk).push()).

Tu peux éventuellement faire un contrôle dans SmartRequest.push() pour vérifier que self.result n'est pas vide, sinon il faut échouer.

#35 - 29 juillet 2021 17:59 - Benjamin Dauvergne

- Statut changé de Solution proposée à Solution validée

#36 - 30 juillet 2021 09:51 - Nicolas Roche

- Statut changé de Solution validée à En cours

Merci pour la validation, mais pense devoir encore continuer à travailler ces patchs.

> Je vois un paramètre trigger mais je ne sais pas d'où il sort, coté create_intervention il y a juste uuid.

Je ne comprend pas, pourtant les triggers n'interviennent que seulement dans 0007 ?

> Tu ne peux pas faire .select_for_update(...) en dehors d'une transaction,

J'avoue que j'ai bêtement copié l'exemple trouvé ici :
https://docs.djangoproject.com/fr/2.2/ref/models/querysets/#select-for-update

> on a dit pas d'authent sur le endpoint de notification

Oui, pardon j'ai oublié ça.

> Pour l'instant ne te préoccupe pas de l'ordre des rejeux des notifications

Compris, je retire la tansaction.

> vire service-error/wcs-errror, tu peux juste laisser le système de job gérer les erreurs,

Non, ici on veut pouvoir lancer une première fois le code de façon asynchrone, sans passer par l'interface des Jobs.
Mais mon incompréhension rejoint certainement le point suivant.

> je ne suis pas certain que le statut sur le modèle SmartRequest ait un intérêt

J'y vois un moyen de se prémunir contre le rejeux.
Si smart veut faire avancer la demande dans le statut A, B puis A, d'après moi il s'agira à chaque fois d'une nouvelle demande de mise à jour.
Je crois comprendre que tu ne l'entends pas comme ça ?

> tout le bouzin de push/mise à jour de smart_request j'en ferai une méthode de smart_request,

Oui, je fais ça.

#37 - 30 juillet 2021 10:34 - Benjamin Dauvergne

Nicolas Roche a écrit :

> Merci pour la validation, mais pense devoir encore continuer à travailler ces patchs.
>
> > Je vois un paramètre trigger mais je ne sais pas d'où il sort, coté create_intervention il y a juste uuid.
>
> Je ne comprend pas, pourtant les triggers n'interviennent que seulement dans 0007 ?

Dans 0005 tu envoies :

```
+        notificationUrl = 'update-intervention?uuid=%s' %
+            'notificationUrl': notificationUrl,
```

C'est là l'URL où SMART va faire son trigger, d'ailleurs je m'aperçois que c'est faux, il faut une URL complète vers update-intervention ici; et je n'y vois que le paramètre uuid pas trigger. SMART va appeler cette URL sans la toucher, il faut qu'elle soit utilisable telle quelle sans authentification et sans y ajouter de paramètre, SMART n'a aucune intégration particulière avec Publik.

> Tu ne peux pas faire .select_for_update(...) en dehors d'une transaction,

J'avoue que j'ai bêtement copié l'exemple trouvé ici :
https://docs.djangoproject.com/fr/2.2/ref/models/querysets/#select-for-update

Oui tu as raison, la requête n'étant faite que lors du premier accès ça va marcher, mais je trouve plus rassurant d'avoir la construction de la query dans le bloc atomique.

> Pour l'instant ne te préoccupe pas de l'ordre des rejeux des notifications

Compris, je retire la tansaction.

> vire service-error/wcs-errror, tu peux juste laisser le système de job gérer les erreurs,

Non, ici on veut pouvoir lancer une première fois le code de façon asynchrone, sans passer par l'interface des Jobs.
Mais mon incompréhension rejoint certainement le point suivant.

J'ai toujours pas bien compris pourquoi on fait pas juste job.run() ici (mais un truc doit m'échapper) en étant dans une transaction, jusqu'à la fin de la transaction le job.

PS: et je vois que add_job envoie systématiquement le job au spooler et que ni la fonction spooler ni la commande Django runjob ne vérifient si le job n'a pas déjà réussi :/ Je pense que si il y avait ces vérifications on pourrait juste faire ça :

```
    with atomic():
         # création smart_request
         job = self.add_job(....)
         job.run()
    return {....}
```

> je ne suis pas certain que le statut sur le modèle SmartRequest ait un intérêt

J'y vois un moyen de se prémunir contre le rejeux.

Je ne vois pas ce que ça va faire concernant le rejeu, y a un job par smartrequest, y en aura jamais deux, vu que c'est le job qui fait le boulot il y a peu de risque qu'on joue deux fois la même notification (c'est des jobs, pas des crons qui se lancent tous seuls).

> Si smart veut faire avancer la demande dans le statut A, B puis A, d'après moi il s'agira à chaque fois d'une nouvelle demande de mise à jour.

Je doute que smart ait beaucoup de statuts à nous notifier et qu'ils soient suffisamment proche pour se marcher dessus, si ça arrive on reverra notre copie. Le rejeu coté wcs vers passerelle est plus certain vu la qualité du réseau local chez NFrance (on a des coupures intempestives de connexion tout le temps). Concernant SMART j'ai plus peur qu'il ne rejoue jamais rien en fait et qu'on perde les notifications.

> Je crois comprendre que tu ne l'entends pas comme ça ?

Pour l'instant je dirais qu'on s'en fout, on a pas le code pour ça ici. Une smart request = un job, on fera plus complexe si nécessaire.

#38 - 30 juillet 2021 12:05 - Nicolas Roche

Désolé d'insister, je ne veux pas donner l'impression de jouer sur les mots, c'est juste que c'est toujours pas clair pour moi.

> 'update-intervention?uuid=%s' : SMART va appeler cette URL sans la toucher, il faut qu'elle soit utilisable telle quelle

Ça veut dire que l'on fige par avance les statuts du WF vers lesquels smart pourra sauter ?
Dans ce cas je pourrais en effet inscrire les triggers en dur dans le code du connecteur et renvoyer une URL par statut.
(mais j'introduis de la logique métier dans le connecteur)

> J'ai toujours pas bien compris pourquoi on fait pas juste job.run() ici

Pour moi, si les jobs sont déjà lancés par ailleurs, vu qu'ils sont (de base) protégés par une transaction, alors l'appel synchrone sera mis en attente à cause de cette transaction, ce qui risquerait de le faire tomber en timeout.

https://dev.entrouvert.org/issues/55840#note-6

> y a un job par smartrequest

Oui, il y a un appel à update_intervention_job() par smartrequest .

> y en aura jamais deux

Pour moi, si smart appel 2 fois update-intervention?UUID=aaa&trigger=bbb
avec les mêmes valeurs aaa et bbb , alors il y aura bien 2 occurrences de smartrequest et donc 2 appels à update_intervention_job() .
Donc tu me demandes de ne considérer qu'un unique objet smartrequest pour les mêmes valeur aaa et bbb ?
Excuse-moi d'insister, mais pour moi c'est là que se situe le point d'achoppement.

```
class SmartRequest(models.Model):
  resource = models.ForeignKey(to=WcsReques...
  trigger = models.CharField(unique=True ...
```

> Je doute que smart ait beaucoup de statuts à nous notifier

'validé', 'planifié', 'mise en sécurité', 'abandonné', 'close manque info' et 'terminé'
Je me disais que la demande d'intervention pouvait être planifiée, mise en sécurité puis planifiée à nouveau.
Mais tu dois avoir raison, vu leurs intitulés, les statuts semblent plutôt linéaires et avec des états terminaux définitifs.
(mais ici aussi, logique métier)

> Une smart request = un job, on fera plus complexe si nécessaire.

Donc si smart rappelait update-intervention?UUID=aaa&trigger=bbb avec des valeurs aaa et bbb déjà soumises, alors on renvoie une seconde fois le trigger à WCS, mais sans noter quelque-part dans le connecteur qu'on l'a envoyé ?

#39 - 30 juillet 2021 12:17 - Benjamin Dauvergne

Le ven. 30 juil. 2021 à 12:05, <redmine@entrouvert.com> a écrit :

> Désolé d'insister, je ne veux pas donner l'impression de jouer sur les mots, c'est juste que c'est toujours pas clair pour moi.
>
> > 'update-intervention?uuid=%s' : SMART va appeler cette URL sans la toucher, il faut qu'elle soit utilisable telle quelle
>
> Ça veut dire que l'on fige par avance les statuts du WF vers lesquels smart pourra sauter ?
> Dans ce cas je pourrais en effet inscrire les triggers en dur dans le code du connecteur et renvoyer une URL par statut.
> (mais j'introduis de la logique métier dans le connecteur)

Peut-être qu'on a pas la même idée de ce qu'est un trigger sur w.c.s, pour moi c'est juste un post à {{ backoffice_form_url }}/trigger/smart, ça n'a rien à voir avec un statut à ma connaissance, après que seul certains statuts écoutent le trigger via un saut conditionné ou que ce soit écouté globalement via une action globale et que ça change le statut de la demande, ça dépend du workflow effectivement; pour moi on appelle toujours le trigger "smart" et le WF se débrouille avec ça.

> > J'ai toujours pas bien compris pourquoi on fait pas juste job.run() ici
>
> Pour moi, si les jobs sont déjà lancés par ailleurs, vu qu'ils sont (de base) protégés par une transaction, alors l'appel synchrone sera mis en attente à cause de cette transaction, ce qui risquerait de le faire tomber en timeout.
> https://dev.entrouvert.org/issues/55840#note-6

Pas compris, le job est transactionnel, on peut (éventuellement, faut pas faire comme si ce serait systématique) notifier plusieurs fois w.c.s. via l'URL de trigger, il devrat se débrouiller avec ça dans son workflow.

> > y a un job par smartrequest
>
> Oui, il y a un appel à update_intervention_job() par smartrequest .

Pas compris ?

> > y en aura jamais deux

> Pour moi, si smart appel 2 fois update-intervention?UUID=aaa&trigger=bbb
> avec les mêmes valeurs aaa et bbb , alors il y aura bien 2 occurrences de smartrequest et donc 2 appels à update_intervention_job() .

C'est normal ça, c'est pas un souci, ça va pas s'exécuter en parallèle.

> Donc tu me demandes de ne considérer qu'un unique objet smartrequest pour les mêmes valeur aaa et bbb ?

Non, un smart request par appel de SMART à passerelle.

> Excuse-moi d'insister, mais pour moi c'est là que se situe le point d'achoppement.

```
> class SmartRequest(models.Model):
>   resource = models.ForeignKey(to=WcsReques...
>   trigger = models.CharField(unique=True ...
>
```

Je ne comprends pas.

> > Je doute que smart ait beaucoup de statuts à nous notifier
>
> 'validé', 'planifié', 'mise en sécurité', 'abandonné', 'close manque info' et 'terminé'
> Je me disais que la demande d'intervention pouvait être planifiée, mise en sécurité puis planifiée à nouveau.
> Mais tu dois avoir raison, vu leurs intitulés, les statuts semblent plutôt linéaires et avec des états terminaux définitifs.
> (mais ici aussi, logique métier)

C'est pas grave en fait même si on revient en arrière dans le wf du
trigger du mets des conditions :

- if status 'validé' jump to status 'smart validé'
- if status 'panifié' jump to status 'smart planifié'
- etc...

> > Une smart request = un job, on fera plus complexe si nécessaire.
>
> Donc si smart rappelait update-intervention?UUID=aaa&trigger=bbb avec des valeurs aaa et bbb déjà soumises, alors on renvoie une seconde fois le
> trigger à WCS, mais sans noter quelque-part dans le connecteur qu'on l'a envoyé ?

Mais d'où smart ajoute un trigger=bbb, ça sort d'où ? Smart va faire :

```
POST /toulouse-smart/<slug>/update-intervention?uuid=<uuid-de-la-demande>

{
"status": "validé",
}
```

et nous on transmet ça à w.c.s sur l'URL d'un trigger fixe qu'on définit, point. Je ne sais pas d'où vient notre incompréhension mutuelle ici, je pense qu'il faut arrêter de parler de trigger peut-être.

#40 - 30 juillet 2021 15:22 - Nicolas Roche

Merci (vraiment) pour tes réponses, désolé de faire le relou.
Je regroupe les questions pour que ça converge.

Trigger

> je pense qu'il faut arrêter de parler de trigger peut-être.

oups

Nom des triggers

> Mais d'où smart ajoute un trigger=bbb, ça sort d'où ?

https://dev.entrouvert.org/issues/55758
ici, j'ai pensé qu'un trigger était associé à un statut

```
statut smart 'validé
{
  "data": {
     "status":"validé"
  }
}
```

(et en fait non, cf ci-dessous)

Trigger sur actions globales

> Peut-être qu'on a pas la même idée de ce qu'est un trigger sur w.c.s,

Merci, j'ai compris : on parle en fait de déclencheurs au niveau des actions globales du workflow.

Je suis parti là dessus :
https://doc-publik.entrouvert.com/dev/wcs/api-webservices/traitement-d-un-formulaire/#asynchrone

et me suis arrêté à :

> et de la référence à l'identifiant de déclencheur

et ne n'ai pas lu / pas pensé aux actions globales :

> Il est également possible de définir des déclencheurs au niveau des actions globales du workflow, ils pourront alors être appelés quel que soit le statut de la demande.

(du coup je comprend)

Représentation des demandes de mises à jour

Un nouvel objet smart_request par appel de SMART à passerelle.

(comme c'est fait dans le patch, mais sans gérer le trigger qui transitera dans les données)

Gestion des erreurs

- si l'erreur se produit dans un job, alors l'erreur se verra inscrite dans le job.
- si l'erreur se produit lors de l'appel synchrone on renvoie l'erreur et on ne l'inscrit null part
  (là c'est moi qui interprète, tu ne m'as pas encore donné d'indication là dessus, peut-être parce que le point suivant n'est pas encore tranché, ou tout simplement parce que c'est un faux problème).

Jobs

```
    with atomic():
         # création smart_request
         job = self.add_job(....)
         job.run()
    return {....}
```

A vrai dire j'étais aussi parti là dessus mais j'ai fait machine arrière quand j'ai cru comprendre que job.run se trouverait en concurrence avec les jobs lancés par le cron et qu'il pourrait ainsi faire sortir l'appel synchrone à update_intervention en timeout.

Je me suis donc rabattu sur ta première proposition suite à https://dev.entrouvert.org/issues/55840#note-7

> Quand le besoin de "tenter d'abord et passer en job sinon", je préfèrerais que l'appel synchrone soit fait directement par l'appelant avant de basculer en job en cas de pépin (timeout rapide ou autre).

#41 - 31 juillet 2021 01:57 - Nicolas Roche

- Fichier 0007-toulouse_smart-add-update-intervention-endpoint-5523.patch ajouté

- Fichier 0005-toulouse_smart-add-create-intervention-endpoint-5523.patch ajouté

- Statut changé de En cours à Solution proposée

Compris. C'est plus simple comme ça, merci.

#42 - 02 août 2021 10:30 - Benjamin Dauvergne

Rajoute un @atomic autour de create_intervention et update_intervention.

```
        try:
            wcs_request.push()
        except ToulouseSmartErrorRetry:
            raise SkipJob()
```

pas la peine d'inventer une exception pour ce cas unique, fais retourner un booléen par push() à False si ça échoue avec des erreurs de transport, sinon laisse passer l'exception et le code devient :

```
       if not wcs_request.push():
           raise SkipJob
```

#43 - 02 août 2021 15:08 - Nicolas Roche

- Fichier 0007-toulouse_smart-add-update-intervention-endpoint-5523.patch ajouté

- Fichier 0005-toulouse_smart-add-create-intervention-endpoint-5523.patch ajouté

(fait)

#44 - 02 août 2021 16:18 - Nicolas Roche

- Fichier 0007-toulouse_smart-add-update-intervention-endpoint-5523.patch ajouté

- Fichier 0005-toulouse_smart-add-create-intervention-endpoint-5523.patch ajouté

- Fichier 0002-toulouse_smart-add-get-intervention-endpoint-55230.patch ajouté

refait.

- get-intervention demande à smart de lui retourner du json.
- transaction.atomic utilisé comme décorateur.

#45 - 03 août 2021 19:03 - Nicolas Roche

- Fichier 0007-toulouse_smart-add-update-intervention-endpoint-5523.patch ajouté

- Fichier 0005-toulouse_smart-add-create-intervention-endpoint-5523.patch ajouté

revu à nouveau :

- correction sur la recherche de doublon
- url absolue générée dans le champs notificationUrl

#46 - 05 août 2021 16:39 - Benjamin Dauvergne

- Statut changé de Solution proposée à En cours

Nicolas Roche a écrit :

> revu à nouveau :
>
> - correction sur la recherche de doublon
> - url absolue générée dans le champs notificationUrl

Je t'embête une dernière fois.

---

ajouter un paramètre timeout=None à push et le mettre à 10 lors de l'appel synchrone, au pire on ralentit le workflow de 10 secondes

---

un pdb/set_trace qui traîne

```
        smart_request = wcs_request.smart_requests.create(payload=post_data)
        smart_request.status = 'registered'
        smart_request.save()
```

tu peux faire directement .create(status='registered', payload=post_data) tu gagnes 2 lignes

---

```
        if smart_request.push():
            smart_request = SmartRequest.objects.get(id=smart_request.id)
            if smart_request.status != 'sent':
                raise APIError(smart_request.result)
```

ça ne sert à rien de recharger SmartRequest à chaque fois, tous ces appels .get(...) sont inutiles, tu travailles toujours sur le même modèle.

de toute façon ici l'appel synchrone me parait inutile, fais juste add_job et tu renvoies ok.

Dans SmartRequest.push :

- factorise la partie pour obtenir un wcs_api et le code de push() sera plus clair
- je ne mettrai pas de valeur par défaut pour les champs JSON
- je mettrai SmartRequest à null=True
- je l'ai déjà dit mais le champ status ne sert à rien, tant que le job existe et n'est pas en erreur on va réessayer et une smartrequest non exécutée sera visible via .filter(result__isnull=True)

#47 - 06 août 2021 11:53 - Nicolas Roche

- Fichier 0007-toulouse_smart-add-update-intervention-endpoint-5523.patch ajouté

- Fichier 0005-toulouse_smart-add-create-intervention-endpoint-5523.patch ajouté

- Fichier 0004-toulouse_smart-manage-payload-and-exception-in-mock_.patch ajouté

- Statut changé de En cours à Solution proposée

Remarques prises en compte, sauf :

> pas de valeur par défaut pour les champs JSON

J'ai laissé, pour moi c'est https://dev.entrouvert.org/issues/52028

> SmartRequest à null=True

Je ne comprend pas de quel champ il s'agit.

#48 - 06 août 2021 12:14 - Benjamin Dauvergne

Nicolas Roche a écrit :

> Remarques prises en compte, sauf :
>
> > pas de valeur par défaut pour les champs JSON
>
> J'ai laissé, pour moi c'est https://dev.entrouvert.org/issues/52028

Je ne vois pas le rapport, le ticket pointé indique un souci parce qu'on passe une valeur par défaut qui n'est pas une fonction et donc est statique (partagée entre tous les objets instanciés). Ici je dis de ne pas mettre de valeur par défaut, si non-null ça force à poser une valeur si NULL ça mettra NULL qui est facile à requèter plutôt que '{}' qui ne l'est pas.

> > SmartRequest à null=True
>
> Je ne comprend pas de quel champ il s'agit.

Pardon, j'ai écrit un peu vite, c'est SmartRequest.result qui est vide jusqu'à ce qu'on obtienne une réponse, c'est typiquement un truc à mettre à NULL.

#49 - 06 août 2021 13:06 - Nicolas Roche

- Fichier 0007-toulouse_smart-add-update-intervention-endpoint-5523.patch ajouté

- Fichier 0005-toulouse_smart-add-create-intervention-endpoint-5523.patch ajouté

pas de valeur par défaut pour les champs JSON

Compris, merci.

#50 - 06 août 2021 14:17 - Benjamin Dauvergne

- Tracker changé de Bug à Development

- Statut changé de Solution proposée à Solution validée

Ok, go.

#51 - 06 août 2021 14:47 - Nicolas Roche

- Statut changé de Solution validée à Résolu (à déployer)

```
commit 9204ba610d2ed88bb170b8576f2769bcb78462ed (HEAD -> main, wip/55230-smart-create-intervention)
Author: Nicolas ROCHE <nroche@entrouvert.com>
Date:   Mon Aug 2 16:09:36 2021 +0200

    toulouse_smart: add update-intervention endpoint (#55230)

commit 8e4776622059adea4df401d295aee86f74eba919
Author: Nicolas ROCHE <nroche@entrouvert.com>
Date:   Mon Aug 2 16:08:48 2021 +0200

    wcs: allow post_json to ask for json content (#55230)

commit ba6f3523adae2e9d2f1282ecefd749837528dd38
Author: Nicolas ROCHE <nroche@entrouvert.com>
Date:   Fri Jul 9 20:07:48 2021 +0200

    toulouse_smart: add create-intervention endpoint (#55230)

commit 5c6a50af6e330a7275cbfe6381a9fa7ca23d47ec
Author: Nicolas ROCHE <nroche@entrouvert.com>
Date:   Fri Jul 9 20:49:33 2021 +0200

    toulouse_smart: manage payload and exception in mock_response (#55230)

commit ff14f2c0e33d912d950e4e14002b0d1a386927ac
Author: Nicolas ROCHE <nroche@entrouvert.com>
Date:   Fri Jul 9 19:35:15 2021 +0200

    toulouse_smart: allow several queries in mock_response (#55230)

commit 3ae9b864f681fcc97e88bbb7504c0c74fa723196
Author: Nicolas ROCHE <nroche@entrouvert.com>
Date:   Fri Jul 9 20:14:44 2021 +0200

    toulouse_smart: add get-intervention endpoint (#55230)

commit 813bfff74fb1eb83450c2b1844c70f48f324a074
Author: Nicolas ROCHE <nroche@entrouvert.com>
Date:   Fri Jul 9 20:31:40 2021 +0200

    toulouse_smart: manage status code in mock_response (#55230)
```

#52 - 09 août 2021 21:17 - Frédéric Péters

- Statut changé de Résolu (à déployer) à Solution déployée

Fichiers

form-smart.wcs	3,09 ko	09 juillet 2021	Nicolas Roche
workflow-smart.wcs	4,62 ko	09 juillet 2021	Nicolas Roche
0007-toulouse_smart-add-update-intervention-endpoint-5523.patch	8,4 ko	09 juillet 2021	Nicolas Roche
0006-toulouse_smart-add-create-intervention-endpoint-5523.patch	18,5 ko	09 juillet 2021	Nicolas Roche
0005-toulouse_smart-check-payload-in-mock_response-55230.patch	4,98 ko	09 juillet 2021	Nicolas Roche

0004-toulouse_smart-allow-several-queries-in-mock_respons.patch	1,8 ko	09 juillet 2021	Nicolas Roche
0003-toulouse_smart-move-make_id-into-utils-55230.patch	3,41 ko	09 juillet 2021	Nicolas Roche
0002-toulouse_smart-add-get-intervention-endpoint-55230.patch	7,63 ko	09 juillet 2021	Nicolas Roche
0001-toulouse_smart-manage-status-code-in-mock_response-5.patch	1,41 ko	09 juillet 2021	Nicolas Roche
0006-wcs-allow-post_json-to-ask-for-json-content-55230.patch	2,08 ko	19 juillet 2021	Nicolas Roche
0007-toulouse_smart-add-update-intervention-endpoint-5523.patch	13,4 ko	19 juillet 2021	Nicolas Roche
0005-toulouse_smart-add-create-intervention-endpoint-5523.patch	26,2 ko	19 juillet 2021	Nicolas Roche
0004-toulouse_smart-check-payload-in-mock_response-55230.patch	4,98 ko	19 juillet 2021	Nicolas Roche
0003-toulouse_smart-allow-several-queries-in-mock_respons.patch	1,8 ko	19 juillet 2021	Nicolas Roche
0002-toulouse_smart-add-get-intervention-endpoint-55230.patch	7,63 ko	19 juillet 2021	Nicolas Roche
0001-toulouse_smart-manage-status-code-in-mock_response-5.patch	1,41 ko	19 juillet 2021	Nicolas Roche
0007-toulouse_smart-add-update-intervention-endpoint-5523.patch	20,4 ko	26 juillet 2021	Nicolas Roche
0005-toulouse_smart-add-create-intervention-endpoint-5523.patch	27,9 ko	26 juillet 2021	Nicolas Roche
0007-toulouse_smart-add-update-intervention-endpoint-5523.patch	20,6 ko	27 juillet 2021	Nicolas Roche
0005-toulouse_smart-add-create-intervention-endpoint-5523.patch	28 ko	27 juillet 2021	Nicolas Roche
0007-toulouse_smart-add-update-intervention-endpoint-5523.patch	20,7 ko	27 juillet 2021	Nicolas Roche
0005-toulouse_smart-add-create-intervention-endpoint-5523.patch	28,2 ko	27 juillet 2021	Nicolas Roche
0007-toulouse_smart-add-update-intervention-endpoint-5523.patch	19,7 ko	30 juillet 2021	Nicolas Roche
0005-toulouse_smart-add-create-intervention-endpoint-5523.patch	30,6 ko	30 juillet 2021	Nicolas Roche
0007-toulouse_smart-add-update-intervention-endpoint-5523.patch	19,8 ko	02 août 2021	Nicolas Roche
0005-toulouse_smart-add-create-intervention-endpoint-5523.patch	30,1 ko	02 août 2021	Nicolas Roche
0007-toulouse_smart-add-update-intervention-endpoint-5523.patch	19,7 ko	02 août 2021	Nicolas Roche
0005-toulouse_smart-add-create-intervention-endpoint-5523.patch	29,8 ko	02 août 2021	Nicolas Roche
0002-toulouse_smart-add-get-intervention-endpoint-55230.patch	7,4 ko	02 août 2021	Nicolas Roche
0007-toulouse_smart-add-update-intervention-endpoint-5523.patch	19,8 ko	03 août 2021	Nicolas Roche
0005-toulouse_smart-add-create-intervention-endpoint-5523.patch	30,6 ko	03 août 2021	Nicolas Roche
0007-toulouse_smart-add-update-intervention-endpoint-5523.patch	17,5 ko	06 août 2021	Nicolas Roche
0005-toulouse_smart-add-create-intervention-endpoint-5523.patch	31,8 ko	06 août 2021	Nicolas Roche
0004-toulouse_smart-manage-payload-and-exception-in-mock_.patch	5,17 ko	06 août 2021	Nicolas Roche
0007-toulouse_smart-add-update-intervention-endpoint-5523.patch	17,5 ko	06 août 2021	Nicolas Roche
0005-toulouse_smart-add-create-intervention-endpoint-5523.patch	31,8 ko	06 août 2021	Nicolas Roche